WATCHLY

Product Requirements Document (PRD)

Version 0.1 | Last Updated: 10 July 2026 | Working Draft

Build Season 2026 | Prepared by the Product Manager

Group 7

Table of Contents

1. Document Control

Field	Details
Product	Watchly — responsive web-based community safety platform
Document	PRD v0.1 — Working Draft
Prepared By / Owner	Product Management Team
Created / Updated	10 July 2026
Build Season	2026
Location	Google Docs — linked in ClickUp

This PRD is the product source of truth: the problem, target users, scope, requirements, and what "ready to release" means. It is a living document — updated as assumptions are validated and decisions are made. Material changes go through the Decision Log (Section 25) and Revision History (Section 26). Task-level execution lives in ClickUp, not here. The TRD, maintained by Engineering, defines how the product is built; the two documents should be read together.

2. Executive Summary

Watchly is a responsive web-based community safety platform. Residents report local safety incidents through a structured process, corroborate reports they have knowledge of, and follow each incident through a transparent status lifecycle. Community Admins review reports and make verification decisions through human-led review. Verified incidents trigger SMS alerts to community members — because a web app cannot reach people who are not in it, and SMS reaches any phone.

Today, community safety information moves through word-of-mouth, phone calls, WhatsApp groups, and social media — fast and familiar, but with no structured reporting, no verification, and no way to tell credible information from false or outdated alerts. Primary research (Section 4) confirms both the problem and conditional willingness to participate in a corroboration mechanism.

The MVP is the smallest end-to-end experience that tests the core hypothesis: **a structured community safety platform can make local incident reporting more credible, actionable, and transparent than the fragmented alternatives residents use today.** It ships as a responsive web app within the Build Season timeline.

3. Product Vision, Boundaries & Principles

3.1 Vision

To build trusted digital infrastructure for community safety — giving residents credible local safety information, a responsible way to participate in reporting and corroboration, and the context to make better-informed safety decisions. Watchly supports authorized security authorities and emergency services; it does not replace them.

3.2 What Watchly Is Not

This is the canonical boundary list — referenced by Goals (Section 6) and Out-of-Scope (Section 10) rather than restated.

- A replacement for emergency services, law enforcement, or official security authorities
- A social platform for discussing crime, rumors, or disputes
- A system where popularity or corroboration counts decide what is true
- A surveillance tool, or a platform that exposes reporter identities, precise locations, or evidence without safeguards

- A guarantee that every reported incident will be accurately resolved

3.3 Product Principles

The canonical home for the trust model. Everywhere else in this PRD, "per Section 3.3" suffices.

Principle	What It Means	Product Implication
Trust Before Virality	Credibility over engagement; reports are never presented as verified fact before review	UI clearly distinguishes Reported, Under Review, Verified, Resolved, Not Verified
Human Oversight	Corroboration is a supporting trust signal — verification decisions stay with authorized human review, never counts or automation	Admins get a defined review workflow with full incident context
Low-Friction Participation	Core actions require minimal steps and data — but never at the expense of safety or data quality	Every required field must justify its existence
Transparency by Default	Users can see incident status, what happens after submitting, and why actions are or are not available	No ambiguous labels or hidden status changes
Privacy & Data Minimization	Collect only what the core experience requires; no address or occupation	Access controls on identity, membership, location, evidence
Community Relevance	Information scoped to communities users intentionally joined	Feeds, alerts, and navigation keep community context clear
Accountability Through Traceability	Critical actions are recorded	Timestamped Status History with responsible actor
Misuse Prevention by Design	Assume safety features will be misused	Abuse safeguards are MVP requirements, not post-launch work
Realistic Usage Conditions	Lower-end devices, unstable networks, never colour alone	Performance, contrast, labels, states in every design decision
Smallest Coherent Product	Complete end-to-end loop over feature count	New features evaluated against evidence, JTBD, and delivery constraints

4. Problem Statement & Research Evidence

4.1 Problem Statement

Residents in Nigerian communities lack a structured, trustworthy way to report, assess, and follow local safety incidents. Existing channels are fragmented, offer no way to assess credibility, and do not support tracking from report to resolution — residents make safety decisions on incomplete or unverified information. Community administrators face the same gap from the other side: no centralized way to review reports, weigh corroboration, communicate updates, or manage incidents through a transparent lifecycle.

4.2 Evidence

The canonical home for all research figures. Other sections reference, not restate.

Context: Nigerian households experienced an estimated 51.89 million crime incidents between May 2023 and April 2024 (NBS Crime Experience and Security Perception Survey). In June 2026, the Alimosho LGA Chairman publicly warned residents about false security alerts circulating on social media — false safety information is an active, named public concern.

Primary research — 74-respondent survey (directional, not nationally representative; self-selected sample):

Finding	Figure	Implication
Witnessed/experienced a local security incident	~63%	The problem space is real
Encountered false, exaggerated, or outdated alerts	~66%	Access to information is not enough — credibility assessment is the gap
Trust in WhatsApp/social media safety info	3 / 5	Used, but not trusted
Would corroborate a witnessed incident	~50%	Corroboration is viable...
“Depends on the situation”	~49%	...but conditional — privacy, safety, and effort concerns need design attention
Reporting channels	Fragmented mix; some do not report at all	No consistent process exists today

Open responses raised verification, credibility, accountability, evidence, and privacy — trust must be built and communicated, not just alerts delivered faster.

4.3 Open Research Questions

Not answered by this research; carried into future validation: why some residents do not report at all; what conditions drive corroboration willingness; willingness to share location or upload evidence; anonymity preferences; expected usage frequency; trust in admins; whether residents would adopt Watchly alongside existing channels. **Community Admins have not been directly researched — all admin needs are hypotheses (Section 20).**

5. Target Users & Personas

5.1 Roles

Role	Who	Core Need
Resident / Community Member (primary)	Anyone with a legitimate interest in a community's safety — lives, works, studies, owns property, or has family there	Find communities, report and follow incidents, corroborate what they genuinely know, stay informed without privacy exposure
Community Admin (secondary)	Users with admin permissions in specific communities (Admin in one, Member in another)	A consistent workflow to review reports, manage status, and communicate accountably — needs are hypotheses pending research
Platform Admin (operational)	System-level users	Review community requests, screen duplicates, approve/decline, activate, assign initial admin

5.2 Proto-Personas

Derived from survey patterns — to be refined through interviews and usage data.

Persona	Profile	Goals	Pain Points
Safety-Conscious Resident	Gets safety info through informal channels; burned by false alerts before	Know what is happening, judge credibility, follow incidents	Cannot tell what to trust; conflicting info; alerts with no follow-up

Persona	Profile	Goals	Pain Points
Occasional Reporter	Witnesses incidents but does not consistently report	Report fast, share only what is necessary, know it was seen	Unclear channels; high effort; no visibility after submitting
Active Contributor	Regularly shares info, helps neighbors stay informed	Improve info quality, confirm what they know, follow multiple communities	Info scattered across chats; no structured way to add credibility
Community Admin	Trusted user managing incident info	Review efficiently, decide consistently, stay accountable	Hypotheses pending direct research

MVP prioritization: Residents first, then enough Community Admin functionality to sustain human-led review, then minimum Platform Admin functionality for community governance. Law enforcement, emergency responders, and institutional partners are outside MVP scope.

5.3 Jobs to Be Done

#	Role	Job
JTBD-1	Resident	Access relevant, contextualized safety information to make better decisions
JTBD-2	Resident	Assess an incident's credibility via status and corroboration before trusting it
JTBD-3	Resident	Report an incident quickly through a clear process so it gets reviewed
JTBD-4	Resident	Corroborate incidents they genuinely know about
JTBD-5	Resident	Follow an incident's status and receive updates — including SMS when verified
JTBD-6	Resident	Participate across multiple communities through one account
JTBD-7	Admin	Review a report's details, evidence, and corroboration in one place
JTBD-8	Admin	Update status through a defined workflow so residents know where things stand
JTBD-9	Admin	Have actions recorded for a transparent history
JTBD-10	Platform Admin	Review community requests and screen duplicates before approving

These jobs demand one coherent loop, not disconnected features: **Account → Join Community → Feed → Report → Corroborate → Admin Review → Status Update → Alerts (SMS on Verified) → Resolution.**

6. Product & Business Goals

6.1 Product Goals

- Structured reporting — a clear process instead of fragmented conversation
- Transparent status — the five statuses distinguishable at a glance
- Responsible participation — corroboration as a supporting signal (per Section 3.3)
- Structured admin workflow — review, decide, track, with visible history
- Community-relevant information — multi-community membership, scoped feeds and alerts
- Inclusive reach — SMS alerts for verified incidents so no one needs the app open to be warned
- A coherent, testable MVP — deployed, measurable, end-to-end

6.2 Business Goals

- Validate the core opportunity: does structure + corroboration + human verification + transparent status beat the informal alternatives?
- Validate participation: will users actually join, report, corroborate, and follow up?
- Evaluate operational viability of human-led verification
- Build an evidence base — usage data, usability findings, security learnings — for what comes next
- Demonstrate cross-functional delivery within Build Season

6.3 MVP Goal Statement

Deliver and evaluate a functional community safety platform where residents join communities, submit and access structured incident information, corroborate reports, follow transparent status, and receive relevant alerts — while Community Admins review and manage reports through a consistent, human-led workflow.

Goal boundaries: per Section 3.2. The MVP also does not aim to prove nationwide product-market fit, drive revenue, or guarantee the accuracy of user-submitted information.

7. Success Metrics & Success Criteria

Success in Build Season means: a functional, usable product; users completing the core end-to-end experience; the data to evaluate key assumptions. Not adoption scale or revenue. With no historical baseline, numerical targets are provisional and reviewed after usability testing and initial usage.

7.1 North Star Metric

Meaningful Incident Engagement Rate (MIER) — % of active users who complete at least one meaningful safety action (submit a report, corroborate, view incident details, or return to view an update) during the measurement period.

MIER = (Active users with ≥1 meaningful incident action ÷ Total active users) × 100

Watchly creates value when users actively access, contribute to, assess, or follow safety information — not when they merely create accounts.

7.2 Metric Set

Area	Key Metrics	Target
Activation	Account creation completion; community join rate; time to first join. Activated = account created + ≥1 community joined	Observational
Reporting	Report completion rate; abandonment rate; median time to submit; % of reports with evidence	Observational
Corroboration	Participation rate; corroborations per eligible incident; self-corroboration blocked; duplicate corroboration blocked	Safeguards: 100%
Admin Review	Time to first review; review rate; outcome distribution (Verified / Not Verified / Resolved / Under Review); median time to decision	Observational
Follow-Through	Detail view rate; follow-up engagement after status change; in-app alert open rate; SMS delivery success rate	Observational
Community	Avg communities per activated user; request rate; approval rate; duplicate-prevention rate	Observational
Usability	Critical task completion (create account, join, report, corroborate, understand status, admin review)	≥80%; zero unresolved blocking issues; SUS ≥68 where administered
Release Quality	Per NFR Release Quality row (Section 15)	Hard gates

7.3 MVP Success Criteria

The MVP is successful for Build Season if:

- A new user can create an account (name, email, phone, password) and access the product
- Users can discover, join, and switch between multiple communities
- Users can submit a structured report with optional images
- The five statuses are clearly distinguishable; eligible users can corroborate once, never their own report
- Community Admins can review and update status through the approved lifecycle; every change is traceable
- Verified incidents trigger SMS alerts to community members; in-app alerts cover other events
- Platform Admins can review, approve/decline, and activate community requests
- Core analytics events are captured and validated
- Critical workflows pass QA, usability, and security release checks
- The deployed MVP produces sufficient evidence to inform the next iteration

Ownership: PM defines the questions and criteria; Data Analytics defines events, feasibility, and the Measurement Plan; Engineering implements instrumentation; QA verifies critical events fire. Changes to the North Star, activation definition, or release thresholds go through the Decision Log.

8. Product Scope

8.1 Scope Overview

The MVP delivers one complete loop: report an incident, corroborate it, review it, track it to resolution. Every feature in scope exists to make that loop work end-to-end for three roles — Residents/Community Members, Community Admins, and Platform Admins.

8.2 In Scope (MVP)

Account & Access. Registration and login with name, email, phone number, and password. Email verification (subject to feasibility). Basic profile management. Role-based access and permissions.

Community Management. Search and discover communities; join more than one. Request a new community where none exists, with duplicate checks. Platform Admin approval or decline. Per-community roles: a user can be an Admin in one community and a Member in another.

Incident Reporting & Information. Structured reports: category, description, community/location context, incident time. Multiple images as optional evidence. Community-scoped incident feed. Incident detail view with current status.

Corroboration. One-tap “I can confirm this incident” on eligible reports. Self-corroboration blocked; duplicate corroboration blocked. Corroboration count displayed as a supporting trust signal — never an automatic verdict (per Section 3.3).

Incident Review & Status Management. Community Admin review of reports, evidence, and corroboration signals in one place. Status lifecycle: Reported → Under Review → Verified → Resolved, or Reported → Under Review → Not Verified. Timestamped status history for every incident.

Alerts & Updates. SMS alerts to community members when an incident is Verified. In-app alerts for other incident activity and status changes, scoped to communities the user has joined.

Platform Administration. Review, approve, or decline community creation requests. Activate approved communities and assign the initial Community Admin.

Analytics, Security & Quality. Core analytics events for all MVP flows. Authentication and authorization controls; basic misuse-prevention safeguards. Responsive web experience tested across critical user flows.

8.3 Scope Boundary

The MVP prioritizes a complete, usable core experience over feature count. Anything that does not directly support reporting, corroboration, review, status tracking, or product validation is deferred — additions require an explicit Decision Log entry. Deferred items are documented in Section 10 so reviewers know they were considered, not missed.

9. MVP Feature Set & Prioritization

9.1 Approach

Features are prioritized using MoSCoW: **Must Have** (required for the core end-to-end experience), **Should Have** (valuable, can be simplified or deferred), **Could Have** (only after Must Haves are stable), **Won't Have** (see Section 10).

9.2 Feature Prioritization

Feature	Priority	Rationale
Registration & login (name, email, phone, password)	Must	Authenticated participation; phone enables SMS alerts
Community discovery, search & multi-join	Must	Users must find communities before anything else works
Community incident feed	Must	Primary way users access safety information
Incident detail view	Must	Context, status, evidence, corroboration in one place
Structured incident reporting (category, location, description, time)	Must	Core product capability
Incident status visibility	Must	Central to the trust proposition
Corroboration + self/duplicate prevention	Must	Core participation mechanism; integrity safeguards are inseparable from it
Admin review, status management & status history	Must	Human-led verification and traceability
Role-based access control	Must	Users only perform authorized actions
Request new community + Platform Admin approval	Must	Controlled network growth
SMS alerts for Verified incidents	Must	Reaches all users regardless of whether the web app is open; core to being user-centered on a non-native platform
Responsive mobile-first web experience	Must	Realistic usage conditions
Core analytics instrumentation	Must	Required to evaluate the MVP
Multiple image evidence uploads	Should	Can reduce to one image if constrained
In-app alerts feed (non-verified events)	Should	Can simplify to an activity feed
Email verification	Should	Depends on technical feasibility
Basic profile management	Should	Not central to the value proposition
Community switching/filtering	Should	Simple selector is enough for MVP
Admin incident filtering	Should	Efficiency, not capability
Duplicate-community suggestions	Should	Simple name + location matching is enough
Save/bookmark incident	Could	Not required for core loop

Feature	Priority	Rationale
Community safety statistics	Could	Post-validation
Advanced search & filtering	Could	After core feed is stable

9.3 Delivery Rule

All Must Haves are implemented, tested, and stable before any Could Have work begins. Should Haves proceed only where they do not put Must Have workflows at risk. Trade-offs are made by the PM in this order: **User Value** → **Core Validation** → **Safety & Trust** → **Dependencies** → **Effort** — and logged in the Decision Log.

9.4 Critical MVP Path

The MVP is not functionally complete unless these two paths work end-to-end:

> *Create Account → Join Community → View Feed → Submit Report → View Detail → Corroborate → Admin Review → Status Update → User Sees Update (SMS on Verified)*
>
> *Request New Community → Platform Admin Review → Approve/Decline → Community Activated → Requester Becomes Initial Admin (where applicable)*

10. Out-of-Scope MVP

Evaluated and intentionally deferred — documented so reviewers know they were considered, not missed.

Capability	Reason
Emergency service / response integration	Requires partnerships, reliability standards, infrastructure
Native mobile apps	Responsive web validates the core experience first
Email and push alerts	SMS (Verified incidents) and in-app alerts are in scope; other channels deferred
AI categorization, summarization, duplicate detection	Requires validation and technical development
Automated verification	Human oversight is central to the trust model
Predictive policing / crime prediction	High-risk; needs data, governance, ethical safeguards
Nationwide public crime mapping	Coverage, moderation, and location safeguards not in place
CCTV / surveillance integration, facial recognition, biometrics	Privacy, safety, ethical, and regulatory risk
Video, audio, document evidence	Image-only reduces moderation, storage, security complexity
Report dispute/challenge mechanism	Corroboration + admin review covers MVP; revisit later
Direct messaging / community chat	Watchly is not a social platform
Payments / monetization	Validate the product first
Trend dashboards & institutional dashboards	Needs usage data and direct stakeholder research
Identity verification / background checks	Third-party services, privacy implications
Continuous location tracking	Explicitly excluded by design
Advanced governance (admin elections, ownership transfer)	Validate the basic community model first

Scope Change Rule: Nothing gets added because there is spare time. Additions are assessed against Research Evidence → User Value → Core Validation → Safety & Privacy → Dependencies → Effort → Impact on Commitments, approved by the PM, and logged in the Decision Log.

11. Information Architecture

11.1 Sitemap

Public: Landing → Sign Up / Login → Verification / Password Recovery

Resident: Home (Incident Feed + community selector + Report action) | Communities (My Communities, Discover, Details, Join, Request New) | Incidents (Detail, Report, Evidence, Corroborate, Status History) | Alerts | Profile & Settings

Community Admin (same account, shown only in authorized communities): Dashboard (review queue, activity, status summary) | Incident Management (detail, evidence, corroboration, status updates, history) | Community Management (info, members, admins)

Platform Admin (separate, system-level access only): Dashboard | Community Requests (details, requester, duplicate check, approve/decline, activate, assign initial admin)

11.2 Primary Navigation

Home | Communities | Report | Alerts | Profile — bottom nav on mobile with Report prominent; sidebar or top nav on desktop. The final pattern is a Design decision, provided core workflows stay accessible.

11.3 Role-Based Navigation Rules

- Admin actions appear only in communities where the user holds admin permissions.
- Platform Admin functionality is never reachable through community permissions.
- Users with no communities see an empty state directing them to discover.
- Users switch between joined communities without logging out; current community context is always visible.

12. User Journey Overview

12.1 Resident Journey

Stage	Key Actions	Outcome
Account Creation	Register, verify, log in	Access to Watchly
Community Membership	Search, view, join one or more	Access to relevant feeds
Safety Awareness	View feed, switch communities, open details	Contextualized safety information
Reporting	Submit structured report, optional images	Incident created as Reported
Participation	Corroborate eligible incidents	Trust signal recorded
Follow-Through	View status changes, receive SMS (Verified) and in-app alerts	Incident followed to outcome

12.2 Community Admin Journey

Stage	Key Actions	Outcome
Access	Log in, select authorized community	Admin workspace
Identify	Open review queue	Reports needing attention found
Review	Assess details, evidence, corroboration, history	Informed decision basis
Decide	Reported → Under Review → Verified / Not Verified	Outcome communicated; Verified triggers SMS
Close	Verified → Resolved when appropriate	Lifecycle completed
Accountability	System records actor + timestamp per change	Traceable history

12.3 Platform Admin Journey

Stage	Key Actions	Outcome
Review	Open pending requests, check duplicates	Informed decision basis
Decide	Approve or decline	Decision recorded
Activate	Activate community, assign initial admin	Community discoverable

12.4 Key Journey Rules

- Account creation does not require joining a community, but community-specific information requires membership.
- Reporting and corroborating require membership in that community.
- Corroboration and permission rules are defined once in Functional Requirements (Section 14) — they apply across all journeys.
- All status changes are recorded in Status History.

13. Core User Flows

Account. Landing → Sign Up (name, email, phone, password) → Verify → Home/Empty State. Returning: Login → Home. Alternates: invalid credentials retry, password recovery, no-communities empty state.

Discover & Join. Communities → Search → Details → Join → Feed. Not found → Request New Community.

Request Community. Enter Name + State + LGA → duplicate check → match found: shown existing to join, or continue → Submit → Pending → Platform Admin decision → Approved (activated, requester assigned admin where applicable, notified) or Declined (notified).

Report Incident. Report → Select Community → Category → General Location → Description → Date/Time → Optional Images → Review → Submit. System: validate, confirm membership, create incident (status = Reported; occurred_at and created_at stored separately), create initial Status History, store evidence, confirm to user.

Corroborate. Incident Detail → "I can confirm this incident" → checks in order: authenticated? → member of community? → own report? (block) → already corroborated? (block) → record, update count. Corroboration never changes verification status.

Admin Review. Dashboard/Queue → Incident (details + evidence + corroboration + history) → Reported → Under Review → Verified or Not Verified → Verified → Resolved. Every change records incident, new status, actor, timestamp. Verified triggers an SMS alert to community members.

Alerts. In-app feed shows events for joined communities (new incidents, status changes, own-report updates, request outcomes) → tap → linked destination. SMS is reserved for Verified incidents only — keeps SMS high-signal and manages cost.

Platform Admin Review. Dashboard → Pending Requests → Details + requester + duplicates → Approve (activate, assign admin, record, notify) or Decline (record reason, notify).

13.1 Critical Error & Empty States

The MVP must define behaviour for: no communities / no results / no incidents / no alerts / no reports pending review; failed registration, login, submission, or upload; network interruption mid-submission; unauthorized admin access; blocked self- or duplicate corroboration; request pending or declined; incident removed; SMS delivery failure (fallback: the in-app alert always fires).

14. Functional Requirements

Each FR is a user story with acceptance criteria QA can test directly. Detailed tasks and test cases live in ClickUp. Any material change to these requirements goes through PM review and the Decision Log.

Authentication

FR-01: Registration

As a new user, I want to create an account so I can participate.

- Registers with full name, email, phone number, and password; fields validated; duplicate email/phone prevented.
- Password requirements are communicated; clear success/error feedback.
- Phone verification (SMS OTP), if implemented, follows the approved rule (OQ-14).

FR-02: Login / Logout

As a returning user, I want secure access to my account.

- Valid credentials → appropriate home experience; invalid → single clear error, no partial login states.
- Logout ends the session; protected pages become inaccessible.

FR-03: Password Recovery

- Reset flow uses secure verification; passwords are never stored or transmitted in plain text.

Communities

FR-04: Discover & Join

As a user, I want to find and join communities relevant to me.

- Search by name/location; results show enough detail to distinguish similar communities; empty state offers Request New Community.
- Users join multiple communities but never the same one twice; can switch between them; current context always displayed.

FR-05: Request New Community

As a user, I want to request a community that does not exist.

- Required info validated; duplicate check runs before submission and shows possible matches first.
- Valid request → Pending status + confirmation to the requester.

Incident Reporting

FR-06: Submit Report

As a member, I want to report an incident so it can be reviewed.

- Only within joined communities. Required: category, general location, description, occurrence date/time. Images optional.
- Success → incident created with Reported status, occurred_at and created_at stored separately, initial Status History record, confirmation shown.
- Failures give clear feedback and allow retry.

FR-07: Image Evidence

- Optional; approved formats/size/count limits enforced; failed uploads do not block submission.
- Images linked to the correct incident; access follows privacy rules.

Feed & Incident Detail

FR-08: Community Feed

- Members see incident cards (summary + status) for joined communities, with empty/loading/error states.
- No access without membership.

FR-09: Incident Detail

- Displays category, general location, description, occurrence + submission time, current status, evidence, corroboration count, status history.
- Reporter identity is never publicly exposed by default. Status is visually distinct from corroboration.
- Actions reflect the viewer's permissions.

Corroboration

FR-10: Corroborate

As a member with genuine knowledge, I want to confirm an incident.

- Action reads “I can confirm this incident”; available only to authenticated members of that community.
- Self-corroboration blocked. Duplicate corroboration blocked. Both with clear explanations.
- Success recorded with user + timestamp; count updates. Never changes verification status.

Admin Review & Status Management

FR-11: Review Access

- Admins access the review queue and incident details (evidence, corroboration, history) only in authorized communities.
- Non-admins cannot reach admin functionality — enforced server-side.

FR-12: Status Updates

- Valid transitions only: Reported → Under Review → Verified | Not Verified; Verified → Resolved. Invalid transitions blocked.
- Every change updates current_status and creates a Status History record (incident, new status, actor, timestamp), visible to eligible users.

Alerts

FR-13: SMS Alerts (Verified Incidents)

As a community member, I want an SMS when an incident in my community is verified, so I am informed even when I am not in the app.

- Marking an incident Verified triggers an SMS to members of that community.
- SMS contains the minimum necessary detail — no reporter identity, no sensitive location precision.

- Users can opt out; delivery failures do not block the status change; the in-app alert always fires as fallback.
- Sending respects agreed gateway limits and cost controls (OQ-15).

FR-14: In-App Alerts

- Feed of relevant events (new incidents, status changes, own-report updates, request outcomes) for joined communities.
- Each alert links to its destination; empty/loading/error states provided.

Platform Administration

FR-15: Community Request Review

- Platform Admins see pending requests, requester details, and possible duplicates; approve or decline.
- Decisions recorded with reasons where required; requester notified in-app.

FR-16: Activation

- Approved community becomes Active and discoverable; requester assigned initial Community Admin where applicable.
- Memberships and permissions created correctly.

Roles, Profile & Analytics

FR-17: Role-Based Access

- System-level Platform Admin plus community-specific Member/Admin roles; a user may hold different roles in different communities.
- Permissions evaluated per active community context, enforced server-side.

FR-18: Profile

- Users view and edit approved fields. Profile image optional. Full residential address and occupation never required.

FR-19: Analytics

- Core events captured per the approved Event Dictionary (Section 17) with the minimum properties needed for MVP metrics.
- No sensitive personal data in events; instrumentation reviewed by PM, Data, and Engineering before build.

15. Non-Functional Requirements

Area	Requirement
Performance	Mobile-first; usable on lower-end devices and unstable networks; visible loading feedback; upload limits enforced; no unnecessary reloads in core flows
Reliability	Clear errors with recovery paths; failed requests never create duplicate reports, corroborations, memberships, or status changes; current_status always consistent with Status History
Security	Passwords hashed, never plain text; authorization enforced server-side, never by hidden UI alone; admin access scoped per community; inputs and uploads validated and sanitized
Privacy	Collect only what MVP functionality requires; reporter identity never public by default; sensitive data accessible by permission only; analytics carry no PII
Accessibility	Target WCAG 2.1 AA for critical flows; never colour alone for status; clear labels, focus states, validation guidance

Area	Requirement
Compatibility	Chrome + at least one other agreed browser; responsive across agreed mobile/tablet/desktop ranges; tested on at least one representative low-end device
Maintainability	Modular structure; errors logged without exposing secrets; no hardcoded credentials or client-exposed secrets
Release Quality	Zero unresolved Critical/Blocker defects; zero unresolved High-Severity vulnerabilities exposing sensitive data; critical flows pass QA; core analytics events verified

Ownership: Engineering reviews feasibility; Cybersecurity reviews security/privacy; QA translates into test scenarios; PM coordinates trade-offs via the Decision Log.

16. Design & UX Requirements

16.1 Core Requirements

- Mobile-first responsive; core actions easy to find; current community context always visible; multi-community switching without friction.
- Clear loading, success, error, empty, and confirmation states everywhere; empty states offer next actions, not blank pages.
- Statuses visually distinct and consistently presented across cards, detail, alerts, and admin views — with text labels, never colour alone.
- Corroboration must never look like verification. “I can confirm this incident” explains what is being confirmed before submission.
- Reporting flow: short, required vs optional clearly marked, review-before-submit, confirmation with a path to the submitted incident.
- Admin actions visually distinct from resident actions; destructive/high-impact actions require confirmation; review queue reachable without digging through the feed.
- Safety content uses clear, neutral language — no sensational or fear-inducing presentation.
- Consistent terminology for communities, incidents, corroboration, roles, and statuses.

16.2 Design Deliverables

The Product Design Team owns: IA + sitemap, user flows/wireflows, screen inventory, wireframes, design system, high-fidelity responsive screens for critical flows, interactive prototype, all states (empty/loading/error/success/permission), handoff specs, and usability findings — all linked in the project workspace and reviewed against approved scope, requirements, and feasibility.

17. Data & Analytics Requirements

17.1 Core Events

Product Area	Events
Account	sign_up_started/completed, login_completed, logout_completed
Communities	community_search_performed, community_viewed, community_joined, community_switched
Requests	community_request_started/submitted/approved/declined
Feed & Detail	incident_feed_viewed, incident_card_clicked, incident_detail_viewed

Product Area	Events
Reporting	incident_report_started/submitted/abandoned/failed
Evidence	image_upload_attempted/completed/failed
Corroboration	corroboration_attempted/completed/blocked
Review	incident_review_started, incident_status_changed
Alerts	alerts_feed_viewed, alert_opened, sms_alert_sent, sms_alert_failed

Properties (per relevance): user_id, community_id, incident_id, user_role, incident_category, previous/new_status, corroboration_count, evidence_attached, failure_reason, timestamp. **Never:** passwords, addresses, evidence content, reporter identity, free text.

17.2 Critical Funnels

- Activation: Sign Up Started → Completed → Community Joined
- Reporting: Report Started → Submitted
- Corroboration: Detail Viewed → Attempted → Completed/Blocked
- Review: Submitted → Review Started → Status Updated → Outcome
- Community Request: Search → No Match → Request Started → Submitted → Outcome

17.3 Quality & Privacy Rules

Consistent naming; documented trigger per event; no double-firing; the Event Dictionary is the source of truth; metrics have documented formulas; events validated before release. Access to raw data is limited by team responsibility.

Deliverables (Data team owns): Measurement Plan, metric definitions, Event Taxonomy + Dictionary, funnel definitions, dashboards, analytics QA checklist, post-launch MVP analysis.

18. Security, Privacy & Compliance

18.1 Requirements by Area

Area	Requirements
Authentication	Auth required for protected functionality; secure hashing; secure session/token handling; secure recovery; safeguards for repeated failed logins per Cybersecurity recommendation
Authorization	Enforced backend-side; community-scoped admin access; Platform Admin restricted to system-level users; permissions evaluated per active community context
User Data	Data minimization; address/occupation never required; reporter identity never public by default; no PII leakage via analytics, logs, URLs, or error messages
Location	Collect only the location detail needed for useful community information; no continuous tracking; visibility rules defined before implementation (OQ-02)
Evidence	Approved formats/sizes/counts only; files validated; no public unrestricted URLs; access by permission
Corroboration Integrity	No self- or duplicate corroboration; records linked to authenticated users for abuse investigation; count never determines verification
SMS	Phone numbers stored securely; SMS content excludes reporter identity and precise location; consent/opt-out honored; gateway credentials never client-exposed

Area	Requirements
Accountability	All critical admin actions attributable (actor + timestamp); audit history cannot be modified or deleted through normal product functionality

18.2 Compliance

Data handling is reviewed against the Nigeria Data Protection Act and NDPC guidance — SMS consent and phone-number storage are now explicitly in scope for this review. User-facing privacy information must explain: data collected, purposes, evidence access, user rights, and how to raise concerns.

18.3 Release Requirements

Before release: auth/authz controls tested, role restrictions verified, incident + evidence access tested, input/upload validation reviewed, abuse risks assessed, findings documented. Zero unresolved Critical vulnerabilities; zero High-Severity vulnerabilities exposing sensitive data.

Deliverables (Cybersecurity owns): Threat Model, Risk Register, security requirements review, auth/RBAC review, privacy review, upload security requirements, abuse assessment, Security Test Plan, testing report, release recommendation.

19. Technical Dependencies & Constraints

19.1 Dependencies

The MVP depends on: approved schema + ERD; API contracts; finalized role/permission model; approved status lifecycle; community logic; image storage; auth/session solution; SMS gateway (e.g., a Termii/Twilio-class provider); email service (if verification ships); alerts logic; analytics spec; hosting; design handoff; QA environments.

External services (hosting, database, image storage, SMS delivery, email, analytics) must be documented with purpose, free-tier limits, security considerations, and delivery impact — SMS per-message cost and volume limits especially, since this is the one dependency with a direct running cost.

19.2 Constraints

ID	Constraint
TC-01	Delivered within the Build Season timeline
TC-02	Must Haves before Should/Could Haves
TC-03	Responsive web only; no native apps
TC-04	Only technologies the team can realistically build, test, and deploy in time
TC-05	No premature optimization or unnecessary complexity
TC-06	Free-tier/service limits identified before implementation decisions — SMS cost model confirmed before FR-13 is locked

19.3 Technical Review & Deliverables

Before full implementation, Engineering confirms: Must Have feasibility, blockers, schema alignment, API needs for critical flows, permission implementation, status transition enforcement, upload limits, SMS gateway integration approach, analytics feasibility, and deployment constraints. Anything forcing a scope change goes to the PM and the Decision Log.

Deliverables (Technical team owns): TRD, architecture, final schema/ERD, API contracts, auth/authz spec, deployment plan, task estimates, testing strategy, tech-debt log.

20. Assumptions & Dependencies

20.1 Key Assumptions

Product: Residents will value structured safety information alongside informal channels; will create accounts, join (multiple) communities, report when it is simple, corroborate what they know, and understand that corroboration ≠ verification. Community-scoped organization gives enough location relevance without tracking. A responsive web app is sufficient to test the proposition. Users will accept providing a phone number and receiving SMS for verified incidents.

Community Admins (unvalidated — no direct research yet): Suitable admins can be identified, will do the review work, can judge reports from evidence + corroboration, and the workload stays manageable. Requester-as-initial-admin is operationally acceptable.

Platform Admins: A small number can handle manual request review at MVP scale; name + location matching is enough for duplicate detection.

Technical: The stack supports Must Haves in time; secure auth + RBAC is implementable; the schema supports multi-community roles; image handling fits service constraints; SMS gateway integration and costs are manageable at MVP volume; analytics can be instrumented and validated before release.

20.2 Cross-Functional Dependencies

Dependency	From	Blocks
Approved PRD v0.1	PM	All teams
Critical flows + screen inventory	Design	FE, BE, QA, Data, Cyber
Final schema + ERD	Backend	APIs, analytics, QA
API contracts	BE + FE	Integration, QA planning
Role/permission spec	PM + BE + Cyber	BE, FE, QA, Security
Measurement Plan + Event Dictionary	Data + PM	FE, BE, QA
Threat Model + security requirements	Cybersecurity	Product, Engineering, QA
SMS gateway selection + cost model	Backend + PM	FR-13, alerts flow, release criteria
Deployment environment	Technical team	Integration + security testing, release

Assumptions are validated through further research, usability testing, technical spikes, security review, and MVP usage data. Material challenges to assumptions go to the Risk Register or Decision Log.

21. Risks & Mitigation

Risk	Category	Impact	Mitigation
False/malicious reports	Trust & Safety	High	Status labels, authenticated accounts, human review, Status History, abuse safeguards
Corroboration mistaken for verification	Trust	High	Visual + copy separation of corroboration and status (Section 16)
Inconsistent admin verification	Operational	High	Defined workflow + criteria (OQ-04), admin guidance, traceable changes
Inappropriate initial Community Admin	Safety	High	Platform Admin review before activation; permission management retained platform-side

Risk	Category	Impact	Mitigation
Reporter/incident data exposure	Privacy	High	Data minimization, hidden identity, access controls, security testing
Harmful/inappropriate image uploads	Trust & Safety	High	Type/size limits, validation, access rules, reported-content process (OQ-12)
Unauthorized admin access	Security	High	Backend RBAC, permission testing, security review
Corroboration manipulation (duplicate accounts)	Trust & Safety	High	Self/duplicate blocks, authenticated accounts, human verification gate
Users stay on WhatsApp	Adoption	High	Lead with differentiated value: structure, status, verification, follow-through
Outsiders joining communities for sensitive info	Privacy	High	Limit exposed detail, review membership rules (OQ-06), stronger controls later
Precise locations create safety risk	Safety	High	Minimum location detail; visibility rules before implementation
SMS costs exceed budget or gateway is unreliable	Technical / Delivery	High	Restrict SMS to Verified incidents only; confirm cost model before locking FR-13; in-app alert always fires as fallback
Admin workload too high	Operational	Medium	Review queue, simple workflows, monitor review times
Low corroboration participation	Adoption	Medium	Low-friction action, clear purpose, evaluate via usage data
Duplicate/misleading communities	Operational	Medium	Duplicate matching, show existing first, Platform Admin approval
Image handling delays delivery	Delivery	Medium	Strict limits early; reduce to one image if needed
Alert noise	UX	Medium	Limited triggers; SMS reserved for Verified only
Late analytics implementation	Data	Medium	Event Dictionary early; validation in QA scope
Late-discovered technical blockers	Delivery	High	Immediate feasibility review of Must Haves; API contracts early
Team silos / divergent assumptions	Delivery	High	PRD as shared baseline; cross-functional reviews; Decision Log
Scope creep	Delivery	High	MoSCoW enforcement; Critical Path protected; changes logged
Insufficient testing before release	Quality	High	Critical scenarios early; incremental testing; release criteria enforced
Must Haves do not fit the timeline	Delivery	High	Smallest releasable core first; dependency-sequenced work; fast trade-offs

Each material risk gets an owner, status, mitigation, and review date in the Risk Register/ClickUp. Detailed security risks live in the Cybersecurity Risk Register.

22. Roadmap & Release Plan

22.1 Phases

Phase	Focus	Key Outputs
1. Discovery & Definition	Validate problem, define product	Research reports, PRD v0.1, MVP scope, initial schema, Decision Log
2. Design & Technical Planning	Make MVP implementation-ready	Approved flows, implementation-ready designs, TRD, architecture, API contracts, final schema, analytics spec, Threat Model, QA plan
3. Development & Continuous Validation	Build and integrate	Working increments, integrated flows, analytics events, test + security + usability findings
4. Stabilization & Release	Test, deploy, evaluate	Deployed MVP, QA report, security recommendation, analytics validation, MVP feedback, retrospective

22.2 Release Gates

Gate	Criteria
1 — Definition Ready	PRD shared; Must Haves agreed; Critical Path defined; feasibility reviewed; blockers identified
2 — Foundation Working	Auth works; core DB relationships live; discovery + membership work; permission foundations in; FE/BE integrating via agreed APIs
3 — Critical Path Integrated	Join → view → report → corroborate → admin review → status update all work; changes recorded
4 — Feature Complete	All Must Haves implemented incl. SMS alerts; analytics instrumented; security controls in; incremental QA done; no end-to-end blockers
5 — Release Ready	Critical flows pass QA; zero Critical/Blocker defects; zero High-Severity exposure vulnerabilities; analytics validated; deployed + smoke-tested; limitations documented; PM/QA/Eng/Cyber release review complete

22.3 Delivery Sequence

> *PRD + Scope → Flows + Feasibility → Schema + APIs + Permissions → Auth + Communities → Feed + Reporting + Detail → Corroboration + Admin Review + Status → Platform Admin + Alerts (in-app + SMS) + Analytics → Integration/Security/Usability Testing → Fixes + Release*

22.4 Scope Contingency

If delivery falls behind, cut in this order: (1) Could Haves; (2) simplify/defer Should Haves (advanced filtering, richer profiles, multiple images, complex duplicate detection, dashboard extras); (3) simplify Must Have implementations without breaking the Critical MVP Path. The Critical Path itself is only touched if a major technical or safety constraint makes delivery impossible.

Single MVP release, supported by continuous internal builds — no big-bang integration at the end. The PM owns the roadmap; each team owns its delivery plan within it.

23. Team Roles & Ownership

Watchly is delivered by a 16-person cross-functional team. Ownership means leading an area and collaborating with dependent teams — not working alone. Task-level assignments live in ClickUp.

23.1 Team Structure

Role	Size	Primary Ownership	Key Deliverables
Product Manager	1	Direction, scope, priorities, decisions, coordination, release	PRD, roadmap, backlog, Decision Log, Open Questions, release coordination
Product Design	5	UX and interface design	Flows, sitemap, wireframes, design system, high-fidelity screens, prototype, handoff, usability testing, design QA
Frontend Development	3	Client-side implementation	Architecture, components, responsive UI, API integration, role-based UI, analytics instrumentation, states, FE testing
Backend Development	1	Server architecture, data, APIs, logic	Schema/ERD, APIs, auth, RBAC, incident lifecycle, corroboration rules, image handling, SMS + alert logic, deployment support
Data Analytics	3	Measurement and analysis	Measurement Plan, Event Dictionary, metrics, funnels, dashboards, analytics QA, MVP analysis
QA	1	Quality and release readiness	Test strategy/plan/cases, functional + integration + regression testing, defect reports, QA release recommendation
Digital Marketing	1	Positioning and communication	MVP positioning, testing recruitment, communication materials, feedback support
Cybersecurity	1	Security, privacy, misuse risk	Threat Model, Risk Register, reviews (auth, RBAC, privacy, uploads, SMS), security testing, release recommendation

23.2 Ownership Matrix (Condensed)

Deliverable	Accountable	Responsible	Consulted
PRD, scope, backlog, roadmap	PM	PM	All
Flows, IA, UI, prototype	PM	Design	FE, BE, QA, Cyber
Architecture, schema, APIs	BE	BE (+FE on contracts)	PM, QA, Cyber
Auth & RBAC	BE	BE + FE	Cyber, QA
Measurement Plan, Event Dictionary	PM / Data	Data	FE, BE, QA
Analytics instrumentation	Data	FE + BE	QA
Threat Model, security testing	Cyber	Cyber	Eng, QA, PM
Test strategy, functional testing	QA	QA	All
Usability testing	PM	Design	Data, QA
Positioning & test comms	PM	Marketing	Design, Data
Scope changes, final acceptance	PM	PM	All
Release readiness	PM	FE, BE, QA, Cyber	Design, Data

23.3 Working Agreements

- Start when inputs are sufficient — do not wait for perfect documents.
- Raise blockers, risks, and requirement conflicts early.
- No team materially changes product behaviour or MVP scope without PM review.
- The PM does not dictate implementation details that belong to functional expertise.
- Decisions get documented, not left in meetings or chats.

- Critical work is tracked in ClickUp with owner, status, priority, and due date.

24. Open Questions

ID	Question	Owner / Consulted	Status
OQ-01	Exact location fields for creating/requesting a community?	PM, Design, BE	Resolved (DEC-033)
OQ-02	Incident location detail collected — and how much is visible to residents?	PM, Design, Cyber, BE	Resolved (DEC-034)
OQ-03	Are reported incidents visible immediately, or only after review begins?	PM, Design, Cyber	Resolved
OQ-04	Exact criteria for Verified / Not Verified / Resolved?	PM, Cyber, QA	Resolved (DEC-035)
OQ-05	Must admins give a reason for status decisions? Which notes are resident-visible?	PM, Design, BE	Resolved
OQ-06	Can anyone join any active community, or are approval/invite controls needed?	PM, Cyber, Design	Resolved (DEC-037)
OQ-07	What info is required to request a community, and what criteria do Platform Admins use to decide?	PM, Design, BE, Cyber	Resolved
OQ-08	Does every approved requester automatically become initial Community Admin? Who can assign/remove admins?	PM, Cyber, BE	Resolved
OQ-09	MVP incident categories?	PM, Design, Data	Resolved
OQ-10	Image formats, max size, max count?	BE, FE, Cyber, PM	Resolved
OQ-11	Can users edit/withdraw a submitted report? Under what conditions?	PM, Design, BE, Cyber	OPEN
OQ-12	Is a Report Content mechanism required for MVP safety?	PM, Cyber, Design, BE	OPEN
OQ-13	Exact in-app alert triggers?	PM, Design, BE, Data	Resolved
OQ-14	Is phone verification (SMS OTP) required at registration, or is the number collected unverified?	BE, Cyber, PM	Resolved (DEC-032)
OQ-15	Which SMS gateway? Per-message cost, volume limits, sender ID, and delivery reliability in Nigeria?	BE, PM, Cyber	OPEN
OQ-16	SMS consent and opt-out model — opt-in at signup or default-on with opt-out? (NDPA review required)	Cyber, PM, Design	OPEN
OQ-17	Feed ordering — chronological, or special treatment for Verified/Under Review?	PM, Design, Data	OPEN
OQ-18	How long do Resolved/Not Verified incidents stay in the active feed?	PM, Design, Data	Resolved
OQ-19	Is an explicit Follow Incident action needed?	PM, Design, BE	OPEN
OQ-20	Is email verification feasible and required for release?	BE, Cyber, PM	OPEN
OQ-21	Editable profile fields?	PM, Design, BE	OPEN
OQ-22	Essential admin + Platform Admin dashboard content beyond queues?	PM, Design, Data	OPEN
OQ-23	Official QA browser/device/viewport matrix?	QA, FE, PM	OPEN

ID	Question	Owner / Consulted	Status
OQ-24	Analytics tool and dashboard solution?	Data, FE, BE, PM	OPEN
OQ-25	External services for hosting, DB, storage, SMS, email?	BE, FE, Cyber	OPEN
OQ-26	User recruitment process for usability/MVP testing?	PM, Design, Marketing	OPEN

Immediate decision priorities (a short Product Decision Workshop, not a general meeting): location granularity and visibility (OQ-01/02), Reported-incident visibility (OQ-03), verification criteria (OQ-04/05), membership rules (OQ-06), community creation + admin rules (OQ-07/08), categories (OQ-09), upload limits (OQ-10), edit/withdraw (OQ-11), Report Content (OQ-12), alert triggers (OQ-13), and the three SMS/phone questions (OQ-14/15/16) — these now block FR-01 and FR-13.

Lifecycle per question: Open → Under Review → Decision Made → Closed. Decisions update the PRD and affected documents, and land in the Decision Log. Questions that do not block the Critical Path do not delay implementation.

25. Decision Log

ID	Decision	Rationale	Status
DEC-001	Responsive web app for MVP	Deliverable within Build Season	Approved
DEC-002	Registration: full name, email, password	Simple onboarding, authenticated participation	Superseded by DEC-032
DEC-003	Profile image optional; address and occupation never required	Data minimization	Approved
DEC-004	Account creation does not require joining a community	Explore before committing	Approved
DEC-005	Multi-community membership per account	Legitimate multi-location interests	Approved
DEC-006	Roles are community-specific	Supports the multi-community model	Approved
DEC-007	Membership required to report or corroborate	Basic participation boundary	Approved
DEC-008–010	Users request communities; duplicate check first; Platform Admin approval required	Controlled growth, reduced duplicates	Approved
DEC-011	Requester may become initial Community Admin	Practical starting model; safeguards unresolved (OQ-08)	Provisional
DEC-012	Lifecycle: Reported → Under Review → Verified → Resolved / Not Verified	Clear, traceable workflow	Approved
DEC-013	occurred_at stored separately from created_at	When it happened ≠ when it was reported	Approved
DEC-014	current_status + separate Status History (actor, timestamp)	Traceability	Approved
DEC-015–018	Corroboration = “I can confirm this incident”; no self- or duplicate corroboration; count never determines verification	Low-friction participation with integrity protection	Approved
DEC-019	Verification decisions are human-led	Safety-sensitive; no automated verdicts	Approved

ID	Decision	Rationale	Status
DEC-020	Reporter identity never public by default	Privacy and personal safety	Approved
DEC-021–022	Optional image evidence only; no video/audio/documents	Context without complexity	Approved
DEC-023	In-app alerts only	Achievable notification scope	Superseded by DEC-031
DEC-024–026	No emergency-service integration, automated verification, predictive policing, crime mapping, CCTV, or surveillance in MVP	Partnerships, ethics, governance beyond scope	Approved
DEC-027	Critical MVP Path before Should/Could Haves	Coherent end-to-end product first	Approved
DEC-028–030	PRD is a working baseline; material changes logged; teams work concurrently	Execution cannot wait for perfect documents	Approved
DEC-031	Verified incidents trigger SMS alerts to community members; in-app alerts cover all other events	A web app cannot reach users who are not in it; SMS works on any phone — the most inclusive channel for the target market. Restricted to Verified incidents to control cost and keep SMS high-signal.	Approved
DEC-032	Registration collects Name, Email, Phone Number and Password. Phone numbers must be verified via OTP before users can report incidents or corroborate reports.	Phone verification reduces fake accounts, improves trust, and enables reliable SMS alerts for verified incidents.	Approved
DEC-033	Community structure follows State → LGA → Community Name.	Provides a consistent location hierarchy for users, backend, and designers.	Approved
DEC-034	Incident reports display a **general location description**, not precise GPS coordinates.	Protects user privacy while providing enough context for residents.	Approved
DEC-035	Incident verification follows a provisional verification policy documented in the PRD.	Ensures administrators review incidents consistently while allowing refinement after pilot testing.	Provisional
DEC-036	Severity is **not included in the MVP** and will be considered after pilot validation.	Keeps the MVP focused on reporting and verification while reducing implementation complexity.	Approved
DEC-037	Community creation requires Platform Admin approval after duplicate checks.	Prevents duplicate communities and reduces abuse while ensuring trusted community administration.	Approved

Log management: New decisions record ID, date, decision, rationale, owner, consulted teams, status, and affected documents (fuller structure in ClickUp). Superseded decisions are never deleted — the status changes, the replacement links back to the original, and affected requirements, designs, and tests get updated.

26. Revision History

Version	Date	Summary	Owner	Status
v0.1	July 2026	Initial working PRD: target users, goals, MVP scope, requirements, journeys, flows, metrics, roadmap, risks,	Product Management	Working Draft

Version	Date	Summary	Owner	Status
		ownership, open questions, decisions. Includes SMS-alert and phone-number decisions (DEC-031/032).		
v0.2	TBD	Post cross-functional review: feasibility findings, resolved OQs, finalized requirements, scope adjustments	Product Management	Planned
v0.3	TBD	Post implementation/testing: usability, security, analytics, QA findings	Product Management	Planned
v1.0	TBD	Final MVP release baseline: implemented scope, accepted requirements, known limitations, deferred work	Product Management	Planned

Log revisions only when changes materially affect scope, requirements, flows, priorities, success criteria, or delivery — not formatting or wording fixes. Minor versions (v0.2, v0.3, ...) for approved changes during build; v1.0 for the release baseline.